Federated Continual Learning with Adaptive Fuzzy Tiling Activation

Chenyang Ma*
University of Cambridge
cm2196@cam.ac.uk

Yuwei Zhang*
University of Cambridge
yz798@cam.ac.uk

Abstract

Federated continual learning (FCL) resembles the real-world applications of cross-device federated learning (FL), in which data within devices evolve over time as users delete, transfer, and add new data consistently. Existing works on FCL showed that FCL suffers from the central problem from continual learning (CL): *catastrophic forgetting*. This work investigates if the techniques developed in CL are also effective in tackling forgetting in FCL so that we can acquire a generalize solution. Specifically, we develop our own method, **Adaptive Fuzzy Tiling Activation (AFTA)**, based on an existing approach, **Fuzzy Tiling Activation (FTA)**, which introduces natural sparsity to reduce interference. We also combine (A)FTA with two additional CL techniques: knowledge distillation and elastic weight consolidation (EWC), to further alleviate forgetting. Extensive evaluations prove the robustness of our method. Code is available at our GitHub repo.

1 Introduction

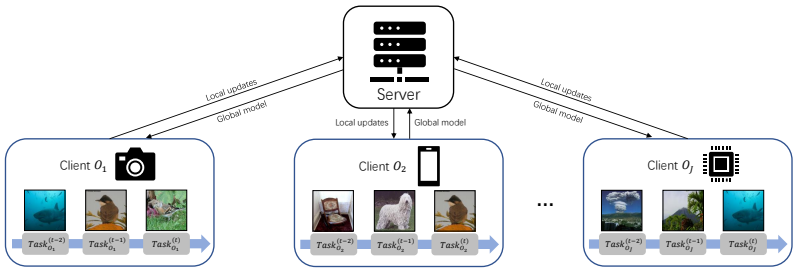


Figure 1: Federated continual learning. Each client is trained on task sequences. The server aggregates local model updates and broadcast the global model to all clients.

Federated learning (FL) [20, 16, 26] allows the training of a global model while keeping datasets localized, which preserves privacy and reduces the amount of data transmission. The first proposed cross-device FL [20] deals with data distributed across individual edge devices (e.g., smartphones, IoT devices). There is a growing interest to deploy cross-device FL because it can utilize rich and diverse user data to conduct meaningful and important downstream tasks such as personalized recommendations, improving user experience, and anomaly detection.

Compared to cross-silo FL, cross-device FL possesses several additional challenges. First, clients (i.e., devices) are massively parallel and can scale up to 10^{10} [11]. Second, clients have intermittent connectivity [23, 35] and suffer from stragglers and dropouts [9, 27], making it difficult for model updates synchronization. Third, clients are heterogeneous in terms of both data distribution [16, 17, 19] and computational resources [14, 10, 3], which constrain the training process and result in poorer model convergence and generalization.

* Indicates equal contribution

While the above challenges of cross-device FL are widely researched, existing literature overlooks the problem of non-stationary data distribution. In the real-world applications of cross-device FL, the data stored on the devices evolve over time as users delete, transfer, and upload new data consistently. Sometimes, data on a device can even be completely updated (consider the case when a smartphone is sold to a new user). We argue that within a single client, the circumstance resembles continual learning (CL) [13, 30], in which the model trains on streaming data (or sequential tasks) without reviewing the previous ones. As shown in Fig. 1, our setting is referred to as federated continual learning (FCL) [34, 5], where we conduct CL with multiple clients trained on task sequences.

Although FCL remains to be under-investigated compared to the widely researched CL [28, 15, 13, 30], works on FCL [34, 5] showed that FCL inherits the same central problem from CL: *catastrophic forgetting*, which occurs when a model trained on a new task forgets or overwrites the knowledge it acquired from previous tasks. This phenomenon is referred to as *interference* [7, 13, 36]. In neural networks (NN), interference occurs because the newly updated weights become sub-optimal for previous tasks. Both Yoon *et al.* [34] and Chaudhary *et al.* [5] mitigated catastrophic forgetting in FCL by developing FL protocols that decompose the network weights into generic global parameters and task-specific parameters; however, their methods do not generalize to various downstream tasks.

The goal of this work is to explore if the techniques developed in the context of CL are also effective in tackling catastrophic forgetting in FCL so that we can acquire a generalized solution. One such technique is **Fuzzy Tiling Activation (FTA)** [24]. FTA tackles forgetting by sparsity, which reduces interference between task sequences by creating more sparse neural representations (i.e., more distributed and efficient encoding of knowledge across tasks), thus leading to less negative interference between task knowledge. We choose and investigate FTA for three specific reasons. First, in contrast to previous works which bring sparsity by learning [21, 6, 31], FTA introduces *natural sparsity* by mapping a scalar to a vector through binning without the need for additional regularization terms or learning parameters. This is particularly desirable for cross-device FL because clients are often resource-constrained. Second, FTA is task agnostic. As it does not require any task-specific information or task boundaries, it can be generalized to induce sparsity and reduce interference between tasks. Third, FTA is model-agnostic and can be easily integrated into model layers, making it suitable for any kind of FL experiment.

The author of FTA stated two promising future research directions for FTA [24]: 1) developing methods to adaptively set the lower and upper bounds of the bin to avoid the problem of *gradient vanishing*; 2) combining FTA with other CL techniques to observe the performance. Following the author's suggestions, we develop a new method to adaptively set the lower and upper bounds of FTA, which we term as **Adaptive Fuzzy Tiling Activation (AFTA)**. Also, we combine FTA with two additional CL techniques including knowledge distillation (KD) [18] and elastic weight consolidation (EWC) [13] to further reduce interference and alleviate catastrophic forgetting.

Overall, the main contributions of this work can be summarized as the following five aspects.

- We adopt **FTA**, an approach to introduce natural sparsity to alleviate catastrophic forgetting, in the context of FCL.
- We propose a novel method, **AFTA**, to adaptively set the lower and upper bounds of the FTA bin.
- We combine the sparsity brought by (A)FTA with two additional CL techniques, **KD** and **EWC**, to further reduce interference and mitigate catastrophic forgetting in FCL.
- Extensive evaluations on two FCL settings prove the effectiveness of FTA and our proposed AFTA. New findings are acquired regarding the effects of combining (A)FTA with KD and EWC.
- We implement our methods under Flower [1], an end-to-end FL framework. Code and checkpoints are made available at our GitHub repo.

2 Problem Formulation: Federated Continual Learning

In CL, the model learns from a sequence of T tasks $\{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, ..., \mathcal{T}^{(T)}\}$. At each task $\mathcal{T}^{(t)}$, $t = 1...T$, the model trains its parameters $\theta^{(t)}$ on the labelled dataset $M^{(t)} = \{x_i, y_i\}_{i=1}^{N_t}$. Note that there is a one-to-one correspondence between $\mathcal{T}^{(t)}$ and $M^{(t)}$ such that the previous $M^{(t-1)}$ datasets are unavailable at time t. The evaluation is incremental. At time t, we evaluate all the previously trained tasks including $\mathcal{T}^{(t)}$: $\{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, ..., \mathcal{T}^{(t-1)}, \mathcal{T}^{(t)}\}$. The problem of catastrophic forgetting refers to the learned parameters $\theta^{(t-1)}$ from previous $t-1$ tasks are forgotten or overwritten by the new knowledge $\theta^{(t)}$.

Extending CL to FCL setting, each client O_j, $j = 1...J$, trains its local model on a privately accessible task sequence $\{\mathcal{T}_{O_1}^{(1)}, \mathcal{T}_{O_2}^{(2)}, ..., \mathcal{T}_{O_J}^{(T)}\}$. Each client progresses through the tasks sequentially and loses access to the data of previous $t-1$ tasks at time t.

Different from existing FCL works [34] focusing on the scenario in which tasks are unrelated and nonidentical across clients at each timestamp, we simplify the FCL setting by coordinating the task switch at a server level so that each client changes tasks at the same time (i.e., at time t, $\mathcal{T}_{O_1}^{(t)} = \mathcal{T}_{O_2}^{(t)} = ... = \mathcal{T}_{O_J}^{(t)}$). This scenario resembles a real-world situation where the data distribution changes over time but in the same way for all clients. For example, in a given region, a seasonal shift results in changes in the image backgrounds taken by users' smartphones, but we always work on the image classification task to distinguish cats from dogs.

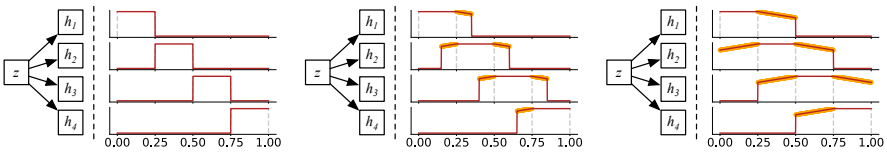
To further clarify, we adopt the domain-incremental learning (Domain-IL) scenario defined in [32], in which the task structure remains the same for all tasks but the input distribution is changing. In Domain IL, the task-ID is not given as part of the inputs (as in Task-IL), but it does not need to be inferred (as in Class-IL). Task A and task B will have different input distributions $P(X_A) \neq P(X_B)$ but the same label distributions $P(Y_A) = P(Y_B)$. The model is single-headed.

3 Natural Sparsity

In this section, we first provide a detailed explanation of FTA in Section. 3.1. Please note that we borrow some notations and equations from the original FTA [24] paper with modifications. Then, we develop a new method to adaptively set the lower and upper bounds for FTA in Section. 3.2. Finally, we explain how to integrate A(FTA) in the context of FL in Section. 3.3.

3.1 Fuzz Tiling Activation

FTA introduces *natural sparsity* to NN by mapping a scalar to a vector with smoothed one-hot encoding. The sparsity claims of FTA is unorthodox compared to previous works in CL which implement sparsity by learning strategies such as weight-space regularization [13, 36] and experience replay [29, 22], but comes with both empirical evidence and theoretical proof.



(a) TA, $k = 4$ (b) FTA, $k = 4, \eta = 0.1$ (c) FTA, $k = 4, \eta = 0.25$

Figure 2: (a) Tiling Activation and (b, c) Fuzzy Tiling Activation with different η. A larger η leads to more leakage of nonzero gradients to the the neighboring bins. We adopt this figure from the original paper of FTA [24] directly. We thank the author Yangchen for making the perfect figure.

Tile Coding Inspired by tile coding, given a scalar z, **Tiling Activation (TA)** converts z to a k-dimensional 1D one-hot encoded vector. We assume z has lower and upper bounds l, u such that $z \in [l, u]$ and $[l, u] \in \mathbb{R}$. We first define the k-dimensional tiling vector v with equal bin spacing ξ, $\xi = (l-u)/k$, $\xi \in \mathbb{R}^+$ as $l < u$. v is:

$$v := (l, l+\xi, l+2\xi, ..., u-2\xi, u-\xi, u) = \text{arange}(l, u+\epsilon, \xi) \tag{1}$$

where ϵ is a very small positive number because arange function is exclusive at the upper bound. We then define an element-wise indicator function, I_+:

$$I_+(w) := \begin{cases} 1, & \text{if } w > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Then, TA is defined as:

$$\vartheta(z) := 1 - I_+ \left(\text{ReLU}(v - z) + \text{ReLU}(z - \xi - v)\right) \tag{3}$$

where $\text{ReLU}(v - z)$ leaves only elements in v greater than z and $\text{ReLU}(z - \xi - v)$ leaves only elements in v less than z by at least one bin spacing ξ. Thus, the output 1D-hot encoded vector $\vartheta(z)$ has: case 1) a 1 in the i-th bin corresponded to the value of z and 0 everywhere else if $v_i < z < v_{i+1}$; and case 2) two 1s in both the $(i-1)$-th and i-th bins if $v_i = z$.

Fig. 2a illustrates TA. We have tiles $k = 4$ and lower and upper bounds $[l, u] = [0, 1]$. Consider two examples: with $z = 0.60$, $\vartheta(0.60) = (0, 0, 1, 0)$; with $z = 0.25$, $\vartheta(0.25) = (1, 1, 0, 0)$.

Leaking the Gradients with Controllable Sparsity TA produces one-hot encoded vector with zero values and derivatives everywhere except the corresponded bin of z, which results in the loss of information during backpropagation. To this end, **Fuzzy Tiling Activation (FTA)** was proposed on top of TA. We first define an element-wise fuzzy indicator function, $I_{\eta,+}$:

$$I_{\eta,+}(w) := I_+(\eta - w) \times w + I_+(w - \eta) = \begin{cases} w, & \text{if } w < \eta \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

Then, FTA is defined as:

$$\vartheta_\eta(z) := 1 - I_{\eta,+}\left(\text{ReLU}(v - z) + \text{ReLU}(z - \xi - v)\right) \tag{5}$$

Fig. 2b and 2c show the effects of FTA. The value of η controls the sparsity. A larger η leaks more gradients to the neighboring bins, resulting in less sparsity (vice versa). It is important to point out that setting $\eta = \xi = (l - u)/k$ (Fig. 2c) is the minimum of η to ensure nonzero gradients everywhere.

A Different Form of Sparsity FTA can be applied to any linear layer in NN, and upscale a linear layer with dimension n to dimension kn, where k is number of tiles. As shown in Fig. 3, FTA is applied to the hidden layer h_1W_2 and outputs $h_2 = \vartheta_\eta(h_1W_2)$.

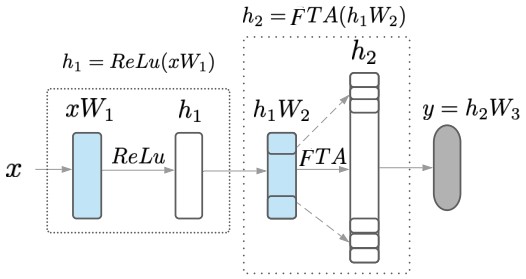


Figure 3: FTA can be applied to any linear layer. We adopt this figure from the original paper of FTA [24] directly. We thank the author Yangchen for making the perfect figure.

Essentially, FTA projects a narrower and denser input layer into a higher-dimensional layer with more sparse encoding. The authors of FTA claim this to be *natural sparsity* as the sparsity is introduced by design instead of learning approaches such as additional regularization terms and losses, pruning, or pre-training [24]. It is crucial to highlight that FTA does not introduce more learning parameters as it solely expands the dimension of the subsequent layer. From another perspective, FTA constructs a more sparse feature space for the learned features.

During the rubuttal stage on OpenReview [1], some reviewers of FTA questioned the sparsity argument of FTA and underscored that it may be unfair to compare *natural sparsity* with sparsity acquired from learning [21, 6, 31]. We do not address this concern here as it is not the focus of this work.

Sparsity Guarantee The authors gave rigorous theoretical proof for FTA regarding its sparsity guarantee. We prompt the readers to refer to Appendix **A.2.3** of the original paper [24].

3.2 FTA with Adaptive Lower and Upper Bounds

An obvious problem of FTA is that if $z \notin [l, u]$, FTA will output zero gradients everywhere, resulting in *gradient vanishing*. Although the author provided empirical evidences [24] that manually setting the lower and upper bounds for FTA can still be effective in CL, we argue that networks with fixed lower and upper bounds for FTA may not perform stably in FCL because clients' local datasets are heterogeneous. Consequently, the magnitude of the gradients within clients' local models varies during backpropagation. Thus, fixed lower and upper bounds worsens the problem of gradient vanishing on some clients, leading to a degradation in the performance of the global model.

To this end, we develop a method to adaptively set the lower and upper bounds l, u during training, which we name as **Adaptive Fuzzy Tiling Activation (AFTA)**. The idea is based on observing the values within the linear layer for which FTA will be applied on. Given a linear layer h with dimension Q, we first predefine a bound set S such that:

$$\forall q \in \{1, 2, ..., Q\}, h_q \in S \tag{6}$$

To make sure Eqn. 6 holds, we can define a bound set S that spans across a wide range of values (e.g., $S = \text{arange}(-25, 25 + \epsilon, 0.5)$). Then, we let $\mu = \text{mean}(h_q), \sigma = \text{std}(h_q)$, and the lower and upper bounds l, u can be determined by:

$$l = \min\{l \in S : l > \mu + 2\sigma\}, \quad u = \max\{u \in S : u < \mu - 2\sigma\}, \tag{7}$$

[1] https://openreview.net/forum?id=zElset1Klrp

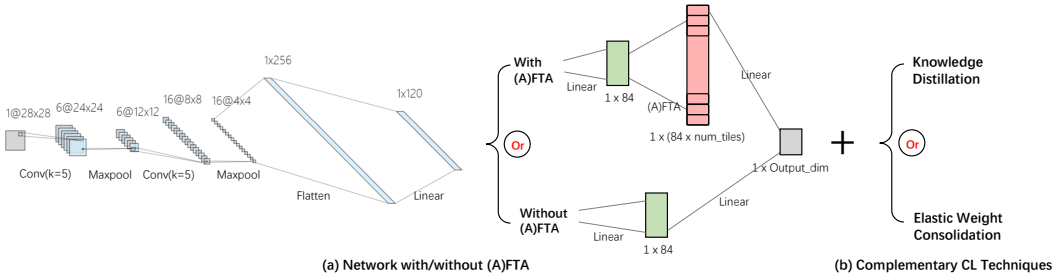


Figure 4: The pipeline. (a) shows the network with/without (A)FTA on the client-side; (b) shows the complementary CL techniques that can be combined with (A)FTA, which is also deployed on the client-side.

such that l is the minimum value in S that is strictly greater than $\mu + 2\sigma$ and u is the maximum value in S that is strictly smaller than $\mu - 2\sigma$.

The reason that we do not set $[l, u] = [\min(h_q), \max(h_q)]$ directly is the author of FTA explicitly pointed out that the model performance is sensitive to the choice of tiling bounds [24]. Thus, our incentive is to encourage the bounds to be more adaptive across clients (with some degrees of freedom within the clients' local models during training across task sequences), and less adaptive within the clients' local models during training across batches (that's why we define a bound set S and use $\mu \pm 2\sigma$ to exclude the most extreme 5% values).

3.3 (A)FTA for Federated Learning

We adopt (A)FTA in the context of FL. For FL strategy, we use FedAvg [20], the first proposed and most basic strategy. In FedAvg, the server aggregates clients' models parameters updates by applying weighted averaging based on the size of clients' local datasets. We do not utilize more advanced FL strategies such as FedProx [16] or FedAdam [26] because exploring the robustness of FL strategies in FCL is not our focus.

Since (A)FTA is task-agnostic and model-agnostic, we deploy (A)FTA on the client-side by integrating (A)FTA to clients' local models, as shown in Fig. 4a. Because our model is a relatively small convolutional neural network (CNN) (more will be explained in Section. 6.2), we insert (A)FTA to the model's last fully connected layer, which is right before the classification head.

4 Continual Learning Techniques to Tackle Catastrophic Forgetting

In this section, we describe two CL techniques which are developed to tackle catastrophic forgetting and are complementary to FTA: knowledge distillation (KD) [8, 18] (Section. 4.1) and Elastic Weight Consolidation (EWC) [13] (Section. 4.2). We combine (A)FTA with KD and EWC, which are also deployed on the client-side (i.e., KD and EWC modify clients' loss function), as shown in Fig. 4b.

4.1 Knowledge Distillation

KD was proposed by Hinton *et al.* [8] and first used to alleviate forgetting in CL by Li *et al.* [18]. The insights of utilizing KD to reduce forgetting can be explained from two perspectives: 1) the teacher model's output prediction probabilities contain valuable implicit knowledge. By learning to match the teacher's probability distribution, the student model preserves information regarding previous tasks and becomes more resilient to forgetting when learning new tasks; 2) to balance the trade-off between fitting the new task and retaining knowledge of previous tasks, we combine Kullback–Leibler divergence (KL div) loss and classification loss (usually cross-entropy loss) with a balancing factor, α. Thus, KD can also be treated as a regularization approach which prevents the model to overfit the knowledge of the new task. With KD, our modified loss term is:

$$\begin{aligned} \mathcal{L}_{\text{KD}} &= (1-\alpha)\mathcal{L}_{\text{CE}} + \alpha\mathcal{L}_{\text{KL}} \\ \mathcal{L}_{\text{KD}} &= (1-\alpha)\left(-\sum_i y_i \log(p_{s_i})\right) + \alpha\left(\sum_i p^T_{t_i} \log\left(\frac{p^T_{t_i}}{p^T_{s_i}}\right)\right) \end{aligned} \tag{8}$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss, $\mathcal{L}_{KL}$ is the KL div loss, α is the balancing factor, T is the temperature coefficient, p_{t_i} and p_{s_i} are the probability outputs scaled by T for class i from the teacher and student models respectively. More specifically, $p^T_{t_i} = \text{softmax}\left(\frac{\text{logits}_{t_i}}{T}\right)$ (same for $p^T_{s_i}$).

4.2 Elastic Weight Consolidation

EWC is a regularization-based technique proposed by Kirkpatrick *et al.* [13]. During the training on a given new task, it mitigates forgetting by regularizing the network's parameters based on their importance for the previously learned tasks. To learn the parameters importance, EWC approximates the true posterior as a Gaussian distribution with the mean given by the model parameters θ^*_A and diagonal precision given by the diagonal of the Fisher information matrix F. When learning a new task B sequentially after the old task A, the loss function of EWC is:

$$\mathcal{L}_{EWC}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i}), \tag{9}$$

where $\mathcal{L}_B(\theta)$ is the loss on task B, λ is a hyperparameter to weigh the penalty inside the loss, and i labels each parameter. Multiple penalties can be enforced to remember more tasks.

5 Implementation Details

We put it in Appendix. A due to space limit. Modification to the Flower classes and self-defined helper functions, as well as the implementation of (A)FTA/EWC/KD are explained.

6 Experiments

In this section, we first introduce the dataset and protocols of our experiments in Section. 6.1. We then report our experimental setup in Section. 6.2 and reveal experimental results in Section. 6.3.

6.1 Dataset

For FCL experiments, we create our own datasets, Permuted- and Split-FEMNIST, from FEMNIST [4] (the federated version of MNIST dataset) and two commonly used benchmark protocols to generate datasets for CL experiments, Permuted- and Split-MNIST [32]. We only use the digits subset and follow the natural partition from FEMNIST based on the writer of the digit.

Permuted-FEMNIST A sequence of 5 tasks is used in Permuted-FEMNIST. Each task is a ten-way classification as in the Permuted-MNIST dataset. The input of each task is generated by applying different pixel-level permutations to the original data. The label and training/test split remain the same. Fig. 5 shows the Permuted-FEMNIST protocol. Note that in our experiments we only use 5 permutations (i.e., 5 sequential tasks) due to limitations of computation resources (the original Permuted-MNIST protocol uses ten).

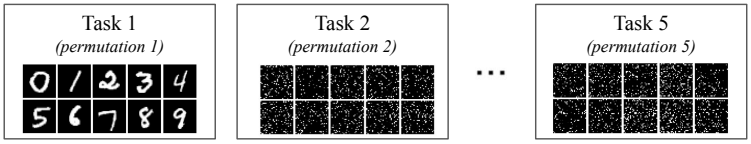


Figure 5: Schematic of Permuted-FEMNIST task protocol. The permutations are the same for all clients.

Split-FEMNIST A sequence of 5 tasks is also used in Split-FEMNIST. The 10 digits of the original FEMNIST dataset for each client is split into 5 tasks, in which each task is a two-way classification. The training/test split remains the same. Fig. 6 shows the Split-MNIST protocol.

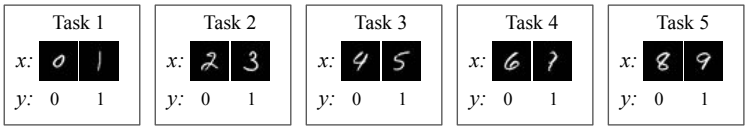


Figure 6: Schematic of Split-FEMNIST task protocol. The split is the same for all clients.

6.2 Experimental setup

Model Architecture We use a small CNN, as shown in Fig. 4a. The input size of all images from Permuted- and Split-FEMNIST is $1 \times 28 \times 28$. For permuted- and Split-FEMNIST, the output dimensions are 10 and 2 respectively.

Comparison Methods For both permuted- and Split-FEMNIST, we use the vanilla CNN (with no sparsity and no CL technique) as our baselines.

Table 1: Performances on Permuted-FEMNIST. **Accuracy** (↑), **Forgetting** ↓. The baseline is in **bold**. The highest Accuracy is in **red**. The lowest forgetting is in **blue**.

Sparsity	CL Tech	$\mathcal{T}^{(1)}$	$\mathcal{T}^{(2)}$	$\mathcal{T}^{(3)}$	$\mathcal{T}^{(4)}$	$\mathcal{T}^{(5)}$	$\mathcal{T}^{(1)*}$	$\mathcal{T}^{(2)*}$	$\mathcal{T}^{(3)*}$	$\mathcal{T}^{(4)*}$	$\mathcal{T}^{(5)*}$	Accuracy	Forgetting
None	None	**18.82**	**22.18**	**22.19**	**21.13**	**44.69**	**53.71**	**33.75**	**35.29**	**36.97**	**44.69**	**25.91**	**15.08**
	KD	23.55	24.58	21.63	24.81	42.74	57.14	35.84	31.63	36.36	42.74	26.96	**13.28**
	EWC	22.20	24.82	21.79	25.39	45.10	57.88	35.55	32.51	36.74	45.10	27.26	13.70
FTA	None	18.79	17.7	18.56	31.51	56.64	76.79	56.84	49.41	56.64	56.64	28.7	30.62
	KD	16.77	19.99	20.36	29.55	54.72	69.37	50.08	41.33	54.42	54.72	27.56	25.71
	EWC	28.29	22.94	23.06	36.56	59.37	79.59	58.49	50.05	66.91	59.37	34.29	28.84
AFTA	None	20.05	25.84	20.77	31.08	46.17	79.06	54.03	50.6	61.82	46.17	30.21	29.55
	KD	28.70	28.45	29.83	36.57	54.35	80.63	47.19	45.74	49.13	54.35	**34.48**	19.83
	EWC	28.56	28.74	29.62	35.65	54.59	80.90	47.68	44.88	49.84	54.59	34.42	20.15

Table 2: Performances on Split-FEMNIST. **Accuracy** (↑), **Forgetting** ↓. The baseline is in **bold**. The highest Accuracy is in **red**. The lowest forgetting is in **blue**.

Sparsity	CL Tech	$\mathcal{T}^{(1)}$	$\mathcal{T}^{(2)}$	$\mathcal{T}^{(3)}$	$\mathcal{T}^{(4)}$	$\mathcal{T}^{(5)}$	$\mathcal{T}^{(1)*}$	$\mathcal{T}^{(2)*}$	$\mathcal{T}^{(3)*}$	$\mathcal{T}^{(4)*}$	$\mathcal{T}^{(5)*}$	Accuracy	Forgetting
None	None	**59.11**	**74.58**	**54.27**	**97.05**	**84.06**	**95.08**	**86.63**	**90.84**	**99.23**	**84.06**	**72.94**	**17.35**
	KD	74.02	61.45	46.95	96.62	85.17	97.56	88.88	70.01	99.52	85.17	75.69	15.39
	EWC	81.45	85.91	51.25	97.68	86.14	96.93	92.25	89.38	98.89	86.14	78.58	12.23
FTA	None	86.66	61.57	44.68	96.42	86.61	98.12	82.90	85.77	99.03	86.61	75.19	15.30
	KD	94.63	69.47	41.78	97.15	87.22	97.51	83.68	60.77	98.28	87.22	78.25	7.44
	EWC	84.15	72.83	45.37	95.55	88.17	97.42	89.01	78.30	98.82	88.17	78.70	13.13
AFTA	None	89.85	81.55	53.2	96.62	80.02	97.75	88.81	90.53	98.48	80.02	**82.68**	10.87
	KD	97.23	84.10	50.26	97.70	81.88	97.91	86.75	77.87	97.99	81.88	82.23	**6.25**
	EWC	86.35	70.40	47.44	96.37	86.48	97.80	87.14	78.74	99.01	86.48	78.83	12.43

Evaluation Metric We evaluate the federated model on the centralized validation set containing all previous tasks and the current task (incremental evaluation as defined in Section. 2). We report the final accuracy of all trained tasks and the best accuracy achieved by each task during the entire training. We also report the performance on two common metrics for CL following CL literature [25, 34, 33]: 1) **Averaged incremental accuracy** (↑) calculates the average accuracy of all tasks at a given time t; 2) **Averaged forgetting** (↓) measures the average difference between the final performance at the last round compared to the best performance across all training rounds of all tasks.

FCL Training and Evaluation Parameters For both permuted- and Split-FEMNIST, we train each of the 5 task for 10 rounds, so there are 50 FCL training rounds. For permuted-FEMNIST, we fix 5 sets of permutation indexes. At each round, we sample 5 clients from 3229 total FEMNIST clients. Each sampled client is trained with Adam [12] optimizer with learning rate of 5e-4, weight decay of 1e-3, local epochs of 10, and batch size of 16. Server-side evaluation uses batch size of 64. We train on one Nvidia Tesla T4 GPU.

(A)FTA, KD, and EWC Parameters For FTA, we set $[l, u] = [-1, 1]$ and number of tiles $k = 10$ as it is one of the recommeded combinations [24]. We set $\eta = \xi = (u - l)/k$ as it is the minimum η to guarantee nonzero gradients everywhere as explained in Section. 3.1. For AFTA, we set $S = [0.01, 0.05, 0, 1, 0.5, 1.0, 5.0, 10.0, 20.0]$ with the negative counterparts as it is the list of reasonable bounds recommended by the author [24]. For KD, we set $\alpha = 0.5$ and $T = 1$ [8] such that cross-entropy loss and KL div loss are balanced with no temperature scaling. For EWC, we follow the default parameters [13].

6.3 Results

We report the quantitative results on Permuted-FEMNIST and Split-FEMNIST in Table. 1 and 2. $\mathcal{T}^{(t)}$ and $\mathcal{T}^{(t)*}$ refer to the final accuracy and the best accuracy of task t respectively. Accuracy is the averaged incremental accuracy across the last three training rounds (i.e., round $48 - 50$). Forgetting is the averaged forgetting.

We also provide the full training curves, which illustrates in detail how our method alleviates the forgetting. They are included in the Appendix. B due to space limit.

7 Discussion

In this section, we explain our observations and findings from the experimental results. We also include limitations and possible future directions of this work.

7.1 Effectiveness of Natural Sparsity by (A)FTA

In both Permuted- and Split-FEMNIST from Table. 1 and 2, we can see that applying FTA or AFTA leads to improvements in averaged incremental accuracy compared to the baseline vanilla CNN, regardless of using other CL techniques or not. This proves the effectiveness of sparsity in reducing interference to mitigate catastrophic forgetting. Also, the best averaged incremental accuracies of both protocols are achieved by AFTA, which implies the robustness of our proposed AFTA to prevent the problem of gradient vanishing of FTA by adaptively setting the lower and upper bounds.

Meanwhile, the averaged forgetting of applying FTA or AFTA is lower than the baseline in the Split-FEMNIST protocol while higher than the baseline in the Permuted-FEMNIST protocol. We hypothesize two possible reasons. First, the permuted task is more difficult so the model is far from convergence within only 10 rounds for each task. The knowledge is not consolidated so the forgetting is quite severe. Second, in addition to introducing sparsity to alleviate forgetting, FTA and AFTA also help the network to reach a much higher best accuracy across the entire training rounds compared to the baseline vanilla CNN, as shown by $\mathcal{T}^{(t)*}$ in Table. 1 and 2. Thus, the averaged forgetting becomes seemingly high as for its calculation, we subtract the performance at the last round from the best performance across all training rounds.

7.2 Effectiveness of Combining (A)FTA with Other CL Techniques

Overall, the results indicate that both KD and EWC are useful and can be combined with (A)FTA. However, we can see that the improvements gained by applying (A)FTA are larger than the improvements gained by applying KD and EWC. This provides another solid evidence that the natural sparsity brought by (A)FTA is crucial in reducing interference and tackling catastrophic forgetting.

Comparing the effects of combining (A)FTA + KD to (A)FTA + EWC, KD turns out to be a more robust method as it yields the lowest average forgetting in all cases and achieves the highest averaged incremental accuracy in most cases.

7.3 Limitations and Future work

Our experiments use the natural partition from FEMNIST based on the writer of the digit, and we did not experiment with partitions of different levels of data heterogeneity. One future work is to explore the robustness of (A)FTA under extreme data heterogeneity using Latent Dirichlet Allocation (LDA) [2] partitions of the FEMNIST dataset. Ideally, AFTA can outperform FTA under high data heterogeneity because the magnitude of the gradients within clients' local models can be divergent (as explained in Section. 3.2).

In addition, since FTA achieves natural sparsity in the feature representation of the model (unlike parameter sparsity by methods such as powerpropagation (powerprop) [31]), the computation and communication cost of the model cannot be reduced. As computation and communication efficiency is a big concern in FL, it will be beneficial to develop methods to reduce these two aspects of FTA.

Last but not least, it will be meaningful to compare FTA with other methods that can bring natural sparsity to the network (e.g., powerprop [31]). We did implement powerprop for FCL; however, in the original paper of powerprop [31], it is coupled with EfficientPackNet (EPN) when applying to CL setting. As we believe either include or exclude EPN will lead to an unfair comparison with FTA, we choose not to include powerprop in the final results.

8 Conclusion

In this work, we explore the use of natural sparsity by Fuzzy Tiling Activation (FTA) in the setting of federated continual learning (FCL). In addition, we propose Adaptive Fuzzy Tiling Activation (AFTA) to deal with the gradient vanishing problem of FTA as clients' local datasets can be heterogeneous. Moreover, we combine two additional CL techniques: knowledge distillation (KD) and elastic weight consolidation (EWC) with (A)FTA, to further alleviate forgetting. Extensive experiments prove the effectiveness of our method and the robustness of combining our proposed AFTA with KD.

References

[1] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. Flower: A friendly federated learning research framework. *arXiv*, abs/2007.14390, 2020.

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 601–608, 2001.

[3] Kallista A. Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *Machine Learning and Systems*, 2019.

[4] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konecný, H. B. McMahan, Virginia Smith, and Ameet S. Talwalkar. Leaf: A benchmark for federated settings. *arXiv*, abs/1812.01097, 2018.

[5] Yatin Chaudhary, Pranav Rai, Matthias Schubert, Hinrich Schütze, and Pankaj Gupta. Federated continual learning for text classification via selective inter-client transfer. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 4789–4799, 2022.

[6] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv*, abs/1907.04840, 2019.

[7] Robert M. French. Catastrophic interference in connectionist networks: Can it be predicted, can it be prevented? In *Advances in Neural Information Processing Systems*, pages 1176–1177, 1993.

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*, 2014.

[9] Samuel Horváth, Stefanos Laskaridis, Mário Almeida, Ilias Leontiadis, Stylianos I. Venieris, and Nicholas D. Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. In *Advances in Neural Information Processing Systems*, pages 12876–12889, 2021.

[10] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, Kaikai Wang, Anthony Shoumikhin, Jesik Min, and Mani Malek. PAPAYA: practical, private, and scalable federated learning. In *Machine Learning and Systems*, 2022.

[11] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, pages 1–210, 2021.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[13] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *arXiv*, abs/1612.00796, 2016.

[14] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. Oort: Efficient federated learning via guided participant selection. In *Symposium on Operating Systems Design and Implementation*, pages 19–35, 2021.

[15] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 3366–3385, 2022.

[16] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Machine Learning and Systems*, 2020.

[17] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. In *International Conference on Learning Representations*, 2021.

[18] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629, 2016.

[19] Wang Lu, Jindong Wang, Yiqiang Chen, Xin Qin, Renjun Xu, Dimitrios Dimitriadis, and Tao Qin. Personalized federated learning with adaptive batchnorm for healthcare. *IEEE Transactions on Big Data*, page 1, 2022.

[20] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[21] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, page 2383, 2018.

[22] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. *arXiv*, abs/1710.10628, 2017.

[23] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *IEEE International Conference on Communications*, pages 1–7, 2019.

[24] Yangchen Pan, Kirby Banman, and Martha White. Fuzzy tiling activations: A simple approach to learning sparse representations online. In *International Conference on Learning Representations*, 2021.

[25] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[26] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

[27] Mónica Ribero, Haris Vikalo, and Gustavo de Veciana. Federated learning under intermittent client availability and time-varying communication constraints. *IEEE Journal of Selected Topics in Signal Processing*, pages 98–111, 2023.

[28] Anthony V. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, pages 123–146, 1995.

[29] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pages 348–358, 2019.

[30] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4535–4544, 2018.

[31] Jonathan Schwarz, Siddhant M. Jayakumar, Razvan Pascanu, Peter E. Latham, and Yee Whye Teh. Powerpropagation: A sparsity inducing weight reparameterisation. In *Advances in Neural Information Processing Systems*, pages 28889–28903, 2021.

[32] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv*, abs/1904.07734, 2019.

[33] Haiyan Yin, Ping Li, et al. Mitigating forgetting in online continual learning with neuron calibration. *Advances in Neural Information Processing Systems*, 34:10260–10272, 2021.

[34] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pages 12073–12086, 2021.

[35] Sixing Yu, Phuong Nguyen, Waqwoya Abebe, Ali Anwar, and Ali Jannesari. SPATL: salient parameter aggregation and transfer learning for heterogeneous clients in federated learning. *arXiv*, abs/2111.14345, 2021.

[36] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995, 2017.

A Implementation Details

In this section, we explain the implementation details. We use the code of L361 Lab 2 as a starting point. We focus on the new codes and modifications made by us within Flower.

A.1 Federated Continual Learning within Flower

We implement a number of classes and functions within Flower to build two FCL settings, following the two protocols that adopt (as explained in Section. 6.1).

For permuted-FEMNIST, we only use the 10 digit classes from FEMNIST. We implement a *PermutedFEMNIST* class, inherited from the original *FEMNIST* class, that takes in a permutation index to initialize. We then implement a function for clients and the server to load the data. We also implement a *PermutedFlowerRayClient* class and overwrite the data loading functions, coupled with a new client generator. For training, we pass the task index through fit configuration based on server rounds to coordinate the task switch of all the clients. For evaluation, we define a function for the server to evaluate the model on all the current and previous tasks that have been seen.

For split-FEMNIST, most modifications follow the same process, except that the data partitions of the 5 tasks are pre-processed and turned into two-way classification tasks.

A.2 (Adaptive) Fuzzy Tiling Activation

The official implementation of FTA is in TensorFlow [2]. There is an unofficial implementation [3] in PyTorch, which we find to be buggy in terms of device resource initialization and modify the majority of it. We define our own FTA class and call it in the *__init__* of our network of type *torch.nn.Module* as an activation function.

For AFTA, since the lower and upper bounds need to be adaptively adjusted during training, we call FTA class inside *forward* instead of *__init__* of our network.

A.3 Knowledge Distillation

We define a custom training function *train_FEMNIST_kd* for the network. For the teacher model, we deepcopy the client's federated model (i.e., the global model received from the server) at the beginning of the round and freeze it by calling *requires_grad_(False)*. The knowledge is distilled into the student model using *torch.nn.KLDivLoss*.

A.4 Elastic Weight Consolidation

We mainly follow the unofficial implementation of EWC [4]. We define a custom *ElasticWeightConsolidation* class containing all utility functions for the training of the network. We make changes in the *_update_fisher_params* function because we encountered an issue that PyTorch is unable to track gradients for the tensors while updating the fisher matrix. We solve this issue by tracking running mean of the log likelihoods instead of storing the probabilities of the current dataloader in a list. We then define a custom training function *train_FEMNIST_ewc*, and call *ewc.forward_backward_update* and *ewc.register_ewc_params* inside.

B Full Training Curves

B.1 Permuted-FEMNIST

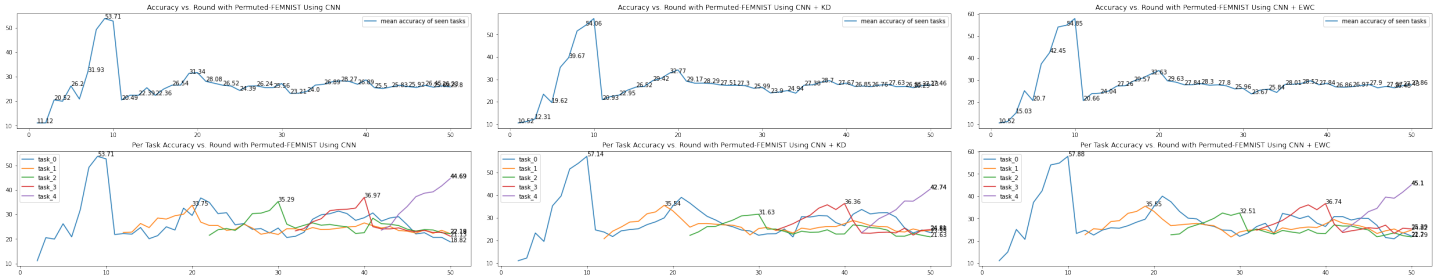
See Fig. 7, 8, and 9.

B.2 Split-FEMNIST

See Fig. 10, 11, and 12.

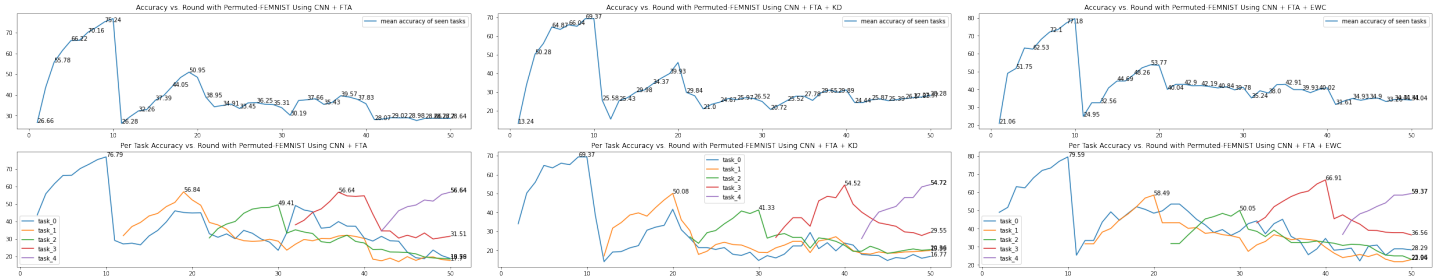
[2] https://github.com/yannickycpan/reproduceRL [3] https://github.com/hwang-ua/fta_pytorch_implementation

[4] https://github.com/shivamsaboo17/Overcoming-Catastrophic-forgetting-in-Neural-Networks



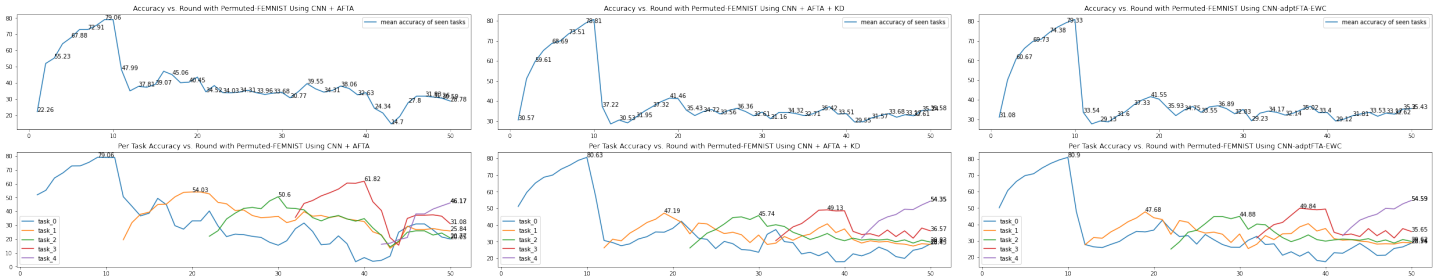
(a) CNN (b) CNN + KD (c) CNN + EWC

Figure 7: Performance on Permuted-FEMNIST Using vanilla CNN, (a) is baseline.



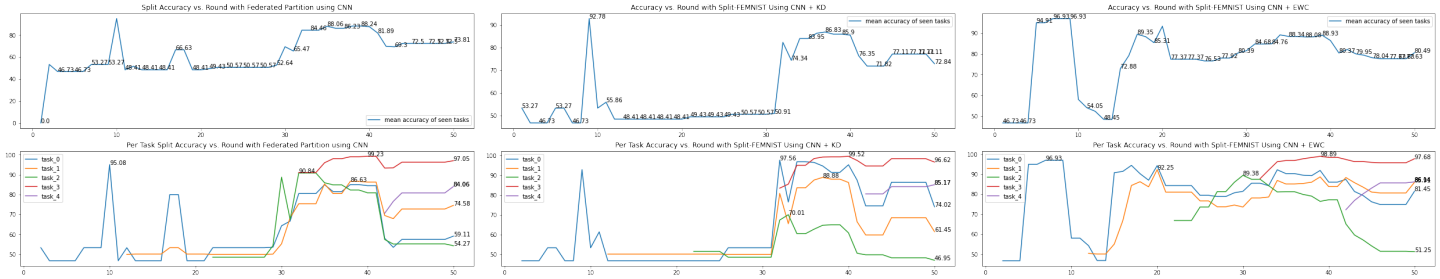
(a) CNN + FTA (b) CNN + FTA + KD (c) CNN + FTA + EWC

Figure 8: Performance on Permuted-FEMNIST Using CNN + FTA.



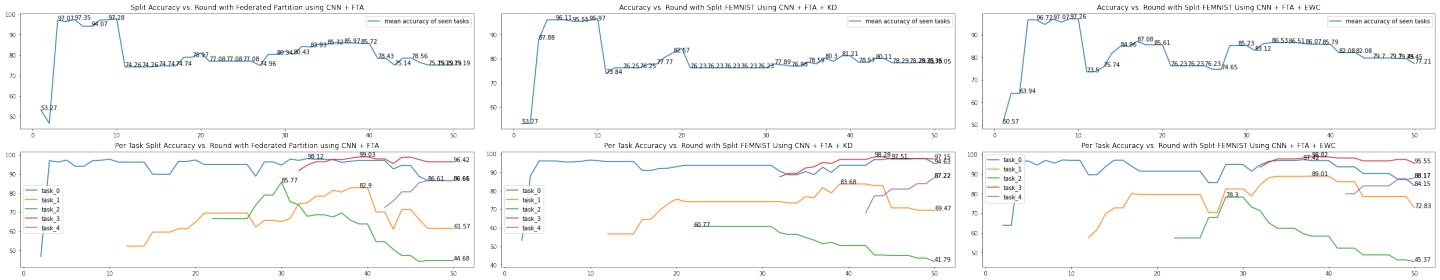
(a) CNN + AFTA (b) CNN + AFTA + KD (c) CNN + AFTA + EWC

Figure 9: Performance on Permuted-FEMNIST Using CNN + AFTA.



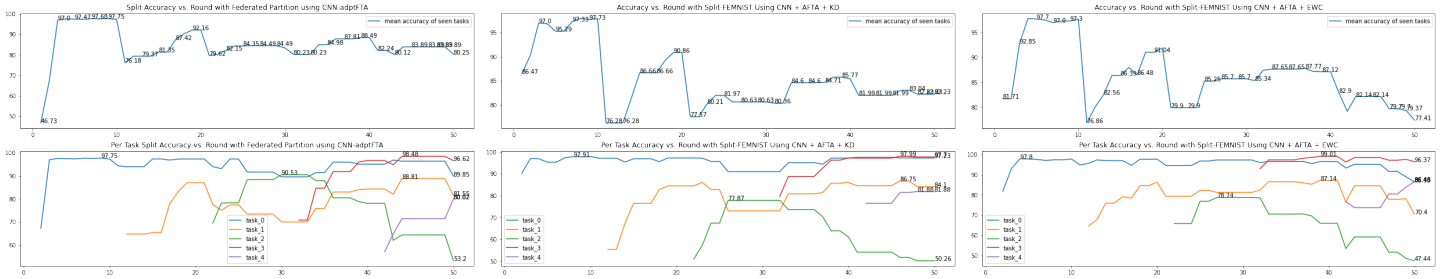
(a) CNN (b) CNN + KD (c) CNN + EWC

Figure 10: Performance on Split-FEMNIST Using vanilla CNN, (a) is baseline.



(a) CNN + FTA (b) CNN + FTA + KD (c) CNN + FTA + EWC

Figure 11: Performance on Split-FEMNIST Using CNN + FTA.



(a) CNN + AFTA (b) CNN + AFTA + KD (c) CNN + AFTA + EWC

Figure 12: Performance on Split-FEMNIST Using CNN + AFTA.